Описание Л-машины

версия 0.9.0, 22 Марта 2004

Важнейшей особенностью Л-машины является строгая типизация всех операций. Преобразование из одного типа данных в другой может быть выполнено только явным образом, с помощью специальных команд.

Таблица 1. Типы данных Л-машины и их размеры

тип	размер, байт
byte	1
integer	4
long	8
single	4, одинарное вещественное IEEE-854
double	8, двойное вещественное IEEE-854
pointer	не специфицируется

При программировании Л-машины явно доступны 12 регистров, по два регистра на каждый тип данных. Регистры именуются r0 и r1. После регистра может стоять суффикс, уточняющий, к какому типу он относится – b (byte), i (integer), l (long), s (single), w (double), p (pointer). Кроме этих существует ряд скрытых регистров, выполняющих специфические функции.

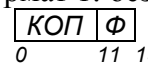
Таблица 2. Регистры Л-машины

регистр	назначение
r0b, r0i, r0l, r0s, r0w, r0p, r1b, r1i, r1l, r1s, r1w, r1p	регистры общего назначения
pc	счетчик инструкций
sp	указатель вершины стека
stack_base	указатель основания стека
task_context	адрес контекста задачи
raised_exception	адрес выброшенного исключения
handlers_table	адрес активной таблицы обработчиков исключений
leave_point	адрес эпилога текущей функции
this_value	адрес активного объекта

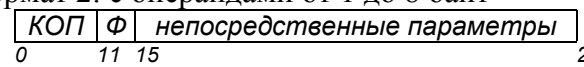
В Л-машине определена 831 команда в 120 семействах операций. Все команды в пределах одного семейства имеют единую семантику, то есть выполняют одну и ту же операцию, но используют разные источники, получатели и типы данных.

Команды имеют переменную длину от 2 до 10 байт. Существует только два формата Л-команд:

формат 1: без операндов



формат 2: с операндами от 1 до 8 байт



В командах Л-машины отсутствуют поля для задания приемников и источников операции. В место этого они жестко задаются в самом коде операции. Операндами могут быть только непосредственные значения, то есть константы, задаваемые в поле кода операции. Большинство арифметических операций являются трехадресными и имеют форму:

```
КОП регистр-источник1, регистр-источник2, регистр-получатель
```

Однако только *регистр-получатель* может быть выбран путем задания кода операции. Для безопасности и облегчения проверки кода, были введены непосредственные операции над переменными, - глобальными и локальными. Важно отметить, что и эти операции строго типизированы.

В самом коде операции только 11 бит используются для задания операции, еще два задают тип адресации непосредственных параметров, оставшиеся биты зарезервированы для использования в дальнейшем.

Таблица 3. Флаги к коде операции (поле Ф)

бит	**интерпретация если не ноль**
11	заменить первый параметр адресом по его значению*
12	заменить второй параметр адресом по его значению*
13-15	не используются

* параметр должен иметь тип integer

Формат операции записывается так, что бы операнд-получатель был последним в списке. Например, `add r0, r1, r0/r1` означает, что семейство операций add берет операнды из регистров r0 и r1, а результат помещает либо в r0, либо в r1.

В тех случаях, когда регистр-источник является переменным, его выбор зависим от выбора регистра получателя. Например, запись вида `agr imm-glob, r0/r1i, r0/r1` означает наличие двух подсемейств инструкции agr: `agr imm-glob, r0i, r0` и `agr imm-glob, r1i, r1`.

Таблица 4. Флаги семейств инструкций

флаг	**описание**
все типы	те источники и получатели, которым не предписан фиксированный тип, могут принимать любой из 6 типов данных
ц. числ. типы	источники и получатели, с не фиксированным типом, могут иметь тип byte, integer и long
integer	источники и получатели, с не фиксированным типом, могут иметь только тип integer
pointer	источники и получатели, с не фиксированным типом, могут иметь только тип pointer
все регистры	для данной инструкции можно выбрать регистр r0 или r1 в качестве источника или получателя
без рег.	регистры не используются и не изменяются
перем. длины	инструкция в данном семействе имеют разную длину, так используют типизированный непосредственный операнд

Таблица 5. Инструкции Л-машины

имя семейства операций	формат операций	флаги	число модификаций	базовый код операции
add (addition)	r0, r1, r0/r1	все типы/все регистры	12	6 (6)
	сложение rD = r0 + r1; rD = {r0, r1}			
sub (subtraction)	r0, r1, r0/r1	все типы/все регистры	12	12 (18)
	вычитание rD = r0 - r1; rD = {r0, r1}			
mul (multiplication)	r0, r1, r0/r1	все типы/все регистры	12	1e (30)
	умножение rD = r0 * r1; rD = {r0, r1}			
div (division)	r0, r1, r0/r1	все типы/все регистры	12	2a (42)
	деление rD = r0 * r1; rD = {r0, r1}			
dec (decrement)	r0/r1	все типы/все регистры	12	4e (78)
	декремент rD = rD – 1; rD = {r0, r1}			
mod (modulus)	r0, r1, r0/r1	ц. числ. типы/все регистры	4	1fe (510)
	модуль rD = rD – 1; rD = {r0, r1}			
and	r0, r1, r0/r1	ц. числ. типы/все регистры	4	202 (514)
	логическое и rD = r0 && r1; rD = {r0, r1}			
or	r0, r1, r0/r1	ц. числ. типы/все регистры	4	206 (518)
	логическое или rD = r0 || r1; rD = {r0, r1}			
lss (less)	r0, r1, r0/r1	все типы/все регистры	12	126 (294)
	арифметический порядок, меньше rD = r0 < r1; rD = {r0, r1}			
grt (greater)	r0, r1, r0/r1	все типы/все регистры	12	132 (306)
	арифметический порядок, больше rD = r0 > r1; rD = {r0, r1}			
lse (less or equal)	r0, r1, r0/r1	все типы/все регистры	12	13e (318)
	арифметический порядок, меньше или равно rD = r0 ≤ r1; rD = {r0, r1}			
gre (greater or equal)	r0, r1, r0/r1	все типы/все регистры	12	14a (330)
	арифметический порядок, больше или равно rD = r0 ≥ r1; rD = {r0, r1}			
eqv (equal)	r0, r1, r0/r1	все типы/все регистры	12	156 (342)
	арифметический порядок, равенство rD = r0 == r1; rD = {r0, r1}			
neq (not equal)	r0, r1, r0/r1	все типы/все регистры	12	162 (354)
	арифметический порядок, неравенство rD = r0 != r1; rD = {r0, r1}			
shl (shift left)	r0, r1, r0/r1	ц. числ. типы/все регистры	4	222 (546)

	сдвиг влево rD = r0 << r1; rD = {r0, r1}			
shr (shift right)	r0, r1, r0/r1	ц. числ. типы/все регистры	4	21e (542)
	сдвиг вправо rD = r0 >> r1; rD = {r0, r1}			
shl (shift left)	r0, imm	integer/все регистры	2	226 (550)
	сдвиг влево rD = r0 << imm; rD = {r0, r1}			
not	r0/r1	ц. числ. типы/все регистры	4	20a (522)
	логическое не rD = !rD ; rD = {r0, r1}			
andb (and binary)	r0, r1, r0/r1	ц. числ. типы/все регистры	4	20e (526)
	побитовое и rD = r0 & r1 ; rD = {r0, r1}			
orb (or binary)	r0, r1, r0/r1	ц. числ. типы/все регистры	4	212 (530)
	побитовое или rD = r0 | r1 ; rD = {r0, r1}			
notb (not binary)	r0, r1, r0/r1	ц. числ. типы/все регистры	4	216 (534)
	побитовое не rD = ~ rD ; rD = {r0, r1}			
xorb (xor binary)	r0, r1, r0/r1	ц. числ. типы/все регистры	4	21a (538)
	побитовое исключающее или rD = r0 ^ r1 ; rD = {r0, r1}			
neg (negation)	r0/r1	все типы/все регистры	12	36 (54)
	арифметическое отрицание rD = -rD ; rD = {r0, r1}			
inc (increment)	r0/r1	все типы/все регистры	12	42 (66)
	инкремент rD = rD + 1 ; rD = {r0, r1}			
incv (increment by value)	r0/r1, imm	все типы/все регистры	12	5a (90)
	инкремент на константу rD = rD + imm ; rD = {r0, r1}			
decv (decrement by value)	r0/r1, imm	все типы/все регистры	12	66 (102)
	декремент на константу rD = rD – imm ; rD = {r0, r1}			
incl (increment local)	imm-loc	все типы/без рег.	6	314 (788)
	инкремент локальной переменной sp[imm-loc] = sp[imm-loc] + 1			
decl (decrement local)	imm-loc	все типы/без рег.	6	31a (794)
	декремент локальной переменной sp[imm-loc] = sp[imm-loc] - 1			
incg (increment global)	imm-glob	все типы/без рег.	6	320 (800)
	инкремент глобальной переменной [imm-glob] = [imm-glob] + 1			
decg (decrement global)	imm-glob	все типы/без рег.	6	326 (806)
	декремент глобальной переменной [imm-glob] = [imm-glob] + 1			

mri (move into register from immediate)	imm, r0/r1	все типы/все регистры/перем. длины	12	72 (114)
	пересылка константы в регистр rD = imm ; rD = {r0, r1}			
mrl (move into register from local)	imm-loc, r0/r1	все типы/все регистры	12	8a (138)
	пересылка локальной переменной в регистр rD = sp[imm-loc] ; rD = {r0, r1}			
mlr (move into a local from a register)	r0/r1, imm-loc	все типы/все регистры	12	96 (150)
	пересылка регистра в локальную переменную sp[imm-loc] = rS ; rS = {r0, r1}			
mll (move into local from local)	imm-loc, imm-loc	все типы/без рег.	6	a2 (162)
	пересылка между локальными переменными sp[imm-loc1] = sp[imm-loc2]			
mgl (move into global from local)	imm-loc, imm-glob	все типы/без рег.	6	a8 (168)
	пересылка локальной переменной в глобальную [imm-glob] = sp[imm-loc]			
mlg (move into local from global)	imm-glob, imm-loc	все типы/без рег.	6	ae (174)
	пересылка глобальной переменной в локальную sp[imm-loc] = [imm-glob]			
mrg (move into register from global)	imm-glob, r0/r1	все типы/все регистры	12	b4 (180)
	пересылка глобальной переменной в регистр rD = [imm-glob] ; rD = {r0, r1}			
mgr (move into global from register)	r0/r1, imm-glob	все типы/все регистры	12	c0 (192)
	пересылка регистра в глобальную переменную [imm-glob] = rS ; rS = {r0, r1}			
mgg (move into global from global)	imm-glob, imm-glob	все типы/без рег.	6	cc (204)
	пересылка между глобальными переменными [imm-glob1] = [imm-glob2]			
mab (move into r0(a) from r1(b))	r1, r0	все типы/без рег.	6	7e (126)
	пересылка между регистрами r0 = r1			
mba (move into r1(b) from r0(a))	r0, r1	все типы/без рег.	6	84 (132)
	пересылка между регистрами r1 = r0			
sbv (store with base and index value)	r0/r1 imm	все типы/все регистры	12	2ec (748)
	сохранение значения по адресу с индексацией по базе [rDp + imm] = rS; rDp = {r0p, r1p}; rS = {r0, r1}			
sta (store from register r0(a))	r0/r1, r0	все типы/все регистры	12	2d4 (724)
	сохранение значения по адресу [rD] = r0; rD = {r0, r1}			
stb (store from register r1(b))	r0/r1, r1	все типы/все регистры	12	2e0 (736)
	сохранение значения по адресу [rD] = r1; rD = {r0, r1}			
objі (new object, size given as immediate)	imm, r0/r1	pointer/все регистры	2	32d (813)
	создание объекта фиксированного размера rD = new (imm); rD = {r0, r1}			

objr (new object, size given in a register)	r0/r1i, r0/r1	pointer/все регистры	2	331 (817)
	создание объекта, размер берется из integer регистра rD = new (rSi); rD = {r0, r1}, rSi = {r0i, r1i}			
objvi (new object with VMLT, size given as immediate)	imm1, imm2, r0/r1	pointer/все регистры	2	32f (815)
	создание объекта фиксированного размера с таблицей описания объекта rD = new (imm2), [rD] = [imm1]; rD = {r0, r1}			
pshl (push local)	imm-loc	все типы/без рег.	6	10e (270)
	помещение локальной переменной в стек push sp[imm-loc]			
pshi (push immediate)	imm	все типы/без рег./перем. длины	6	11a (282)
	помещение константы в стек push imm			
push	r0/r1	все типы/все регистры	12	f6 (246)
	помещение регистра в стек push rS; rS = {r0, r1}			
pop	r0/r1	все типы/все регистры	12	102 (258)
	извлечение регистра из стека push rD; rD = {r0, r1}			
pshg (push global)	imm-glob	все типы/без рег.	6	114 (276)
	помещение глобальной переменной в стек push [imm-glob]			
spadj (sp adjust)	imm		1	338 (824)
	изменение значения вершины стека sp = sp + imm			
swap	r0, r1	все типы/без рег.	6	120 (288)
	обмен значений между регистрами tmp = r1, r1 = r0, r0 = tmp			
jump	imm		1	32c (812)
	безусловный переход jump [imm]			
jt (jump if true)	r0/r1, imm	все типы/все регистры	12	d2 (210)
	условный переход, если r0/r1 не равен нулю if rS then jump [imm]; rS = {r0, r1}			
jf (jump if false)	r0/r1, imm	все типы/все регистры	12	de (222)
	условный переход, если r0/r1 равен нулю if not rS then jump [imm]; rS = {r0, r1}			
jls* (jump if less)	r0/r1, imm1, imm2	integer/все регистры	2	ea (234)
	условный переход, если r0/r1 меньше заданной константы if rS < imm1 then jump [imm2]; rS = {r0, r1}			
jle* (jump if less or equal)	r0/r1, imm1, imm2	integer/все регистры	2	ec (236)
	условный переход, если r0/r1 меньше или равен заданной константе if rS <= imm1 then jump [imm2]; rS = {r0, r1}			
jgr* (jump if greater)	r0/r1, imm1, imm2	integer/все регистры	2	ee (238)
	условный переход, если r0/r1 больше заданной константы if rS > imm1 then jump [imm2]; rS = {r0, r1}			

jge* (jump if greater or equal)	r0/r1, imm1, imm2	integer/все регистры	2	f0 (240)
	условный переход, если r0/r1 больше или равен заданной константе if rS >= imm1 then jump [imm2]; rS = {r0, r1}			
jeq* (jump if equal)	r0/r1, imm1, imm2	integer/все регистры	2	f2 (242)
	условный переход, если r0/r1 равен заданной константе if rS == imm1 then jump [imm2]; rS = {r0, r1}			
jne* (jump if not equal)	r0/r1, imm1, imm	integer/все регистры	2	f4 (244)
	условный переход, если r0/r1 равен заданной константе if rS == imm1 then jump [imm2]; rS = {r0, r1}			
byt (cast to byte)	r0/r1, r0b	все типы/все регистры	12	16e (366)
	преобразование в тип byte r0b = (byte) rS; rS = {r0, r1}			
int (cast to integer)	r0/r1, r0i	все типы/все регистры	12	17a (378)
	преобразование в тип integer r0i = (integer) rS; rS = {r0, r1}			
lng (cast to long)	r0/r1, r0l	все типы/все регистры	12	186 (390)
	преобразование в тип long r0l = (long) rS; rS = {r0, r1}			
flt (cast to float)	r0/r1, r0f	все типы/все регистры	12	192 (402)
	преобразование в тип float r0f = (float) rS; rS = {r0, r1}			
dbl (cast to double)	r0/r1, r0w	все типы/все регистры	12	19e (414)
	преобразование в тип double r0w = (double) rS; rS = {r0, r1}			
byt (cast to byte)	r0/r1, r1b	все типы/все регистры	12	1b6 (438)
	преобразование в тип byte r1b = (byte) rS; rS = {r0, r1}			
int (cast to integer)	r1/r1, r1i	все типы/все регистры	12	1c2 (450)
	преобразование в тип integer r0i = (integer) rS; rS = {r0, r1}			
lng (convert to long)	r1/r1, r1l	все типы/все регистры	12	1ce (462)
	преобразование в тип long r0l = (long) rS; rS = {r0, r1}			
flt (cast to float)	r0/r1, r1f	все типы/все регистры	12	1da (474)
	преобразование в тип float r1f = (float) rS; rS = {r0, r1}			
dbl (cast to double)	r0/r1, r1w	все типы/все регистры	12	1e6 (486)
	преобразование в тип double r1w = (double) rS; rS = {r0, r1}			
alr4 (address from local and register, scale 4)	imm-loc, r0/r1i, r0/r1	pointer/все регистры	2	2bc (700)
	относительная адресация через локальную переменную и индекс с масштабным индексом 4 rDp = [sp[imm-loc]] + rSi*4; rD = {r0p, r1p}; rSi = {r0i, r1i}			
alr8 (address from	imm-loc, r0/r1i, r0/r1	pointer/все регистры	2	2be (702)

	относительная адресация через локальную переменную и индекс с масштабным индексом 8 rDp = [sp[imm-loc]] + rSi*8; rD = {r0p, r1p}; rSi = {r0i, r1i}			
agr4 (address from global and register, scale 4)	imm-glob, r0/r1i, r0/r1	pointer/все регистры	2	2c4 (708)
	относительная адресация через глобальную переменную и индекс с масштабным индексом 4 rDp = [[imm-glob]] + rSi*4; rD = {r0p, r1p}; rSi = {r0i, r1i}			
agr8 (address from global and register, scale 8)	imm-glob, r0/r1i, r0/r1	pointer/все регистры	2	2c6 (710)
	относительная адресация через глобальную переменную и индекс с масштабным индексом 8 rDp = [[imm-glob]] + rSi*8; rD = {r0p, r1p}; rSi = {r0i, r1i}			
arr4 (address from register and register, scale 4)	r0/r1, r0/r1i, r0/r1	pointer/все регистры	2	2cc (716)
	относительная адресация через регистр и индекс с масштабным индексом 4 rDp = [rDp]+ rSi*4; rD = {r0p, r1p}; rSi = {r0i, r1i}			
arr8 (address from register and register, scale 8)	r0/r1, r0/r1i, r0/r1	pointer/все регистры	2	2ce (718)
	относительная адресация через регистр и индекс с масштабным индексом 8 rDp = [rDp]+ rSi*8; rD = {r0p, r1p}; rSi = {r0i, r1i}			
vlr4 (value addressed by local and register, scale 4)	imm-loc, r0/r1, r0/r1	все типы/все регистры	12	240 (576)
	взятие значения адресованного локальной переменной и индексом с масштабным индексом 4 rDp = [[sp[imm-loc]]+ rSi*4]; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
vlr8 (value addressed by local and register, scale 8)	imm-loc, r0/r1, r0/r1	все типы/все регистры	12	24c (588)
	взятие значения адресованного локальной переменной и индексом с масштабным индексом 8 rDp = [[sp[imm-loc]]+ rSi*8]; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
vgr4 (value addressed by global and register, scale 4)	imm-glob, r0/r1, r0/r1	все типы/все регистры	12	270 (624)
	взятие значения адресованного глобальной переменной и индексом с масштабным индексом 4 rDp = [[[imm-glob]]+ rSi*4]; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
vgr8 (value addressed by global and register, scale 8)	imm-glob, r0/r1, r0/r1	все типы/все регистры	12	27c (636)
	взятие значения адресованного глобальной переменной и индексом с масштабным индексом 8 rDp = [[[imm-glob]]+ rSi*8]; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
enter	imm1, imm2		1	2 (2)
	пролог функции. Настраивает контекст для выполнения подпрограммы leave_point = [imm1], sp = sp + imm2			
leave			1	3 (3)
	уничтожение текущего контекста и возвращение в контекст вызывающей функции			
fcall (function call)	imm		1	334 (820)
	вызов функции по непосредственному адресу jump [imm]			
vcall (virtual method call)	imm		1	336 (822)
	вызов виртуального метода, адрес соответствующего объекта берется из вершины стека (без извлечения). Значение imm служит смещением в таблице виртуальных методов pop obj, jump [[obj] + imm + 1]			

icall (interface method call)	imm, imm		1	337 (823)
	вызов интерфейсного метода, адрес соответствующего объекта берется из вершины стека (без извлечения)			
funct* (function call)	imm		1	333 (819)
	вызов функции без передачи контекста			
ncall* (native call)	imm		1	335 (821)
	вызов функции с передачей параметров согласно ABI данной системы.			
vli (value addressed by local and immediate)	imm-loc, imm, r0/r1	все типы/все регистры	12	228 (552)
	взятие значения адресованного локальной переменной и индексом - константой rD = [[sp[imm-loc] + imm]; rD = {r0, r1}			
vlr (value addressed by local and register)	imm-loc, r0/r1, r0/r1	все типы/все регистры	12	234 (564)
	взятие значения адресованного локальной переменной и индексом rD = [[sp[imm-loc]]+ rSi]; rD = {r0, r1}; rSi = {r0i, r1i}			
vri (value addressed by register and immediate)	r0/r1, imm, r0/r1	все типы/все регистры	12	288 (648)
	взятие значения адресованного регистром и индексом - константой rD = [[rSp]]+ imm]; rD = {r0, r1}; rSp = {r0p, r1p}			
vrr (value addressed by register and register)	r0/r1, r1/r0, r0/r1	все типы/все регистры	12	294 (660)
	взятие значения адресованного регистром и индексом rD = [[rSp]]+ rSi]; rD = {r0, r1}; rSp = {r0p, r1p}; rSi = {r1i, r0i}			
vrr4 (value addressed by register and register, scale 4)	r0/r1, r1/r0, r0/r1	все типы/все регистры	12	2a0 (672)
	взятие значения адресованного регистром и индексом с масштабным индексом 4 rD = [[rSp]]+ rSi*4]; rD = {r0, r1}; rSp = {r0p, r1p}; rSi = {r1i, r0i}			
vrr8 (value addressed by register and register, scale 8)	r0/r1, r1/r0, r0/r1	все типы/все регистры	12	2ac (684)
	взятие значения адресованного регистром и индексом с масштабным индексом 8 rD = [[rSp]]+ rSi*8]; rD = {r0, r1}; rSp = {r0p, r1p}; rSi = {r1i, r0i}			
ali (address from local and immediate)	imm-loc, imm, r0/r1	pointer/все регистры	2	2b8 (696)
	относительная адресация через локальную переменную и индекс – константу rDp = [sp[imm-loc]] + imm; rDp = {r0p, r1p}			
alr (address from local and register)	imm-loc, r0/r1i, r0/r1	pointer/все регистры	2	2ba (698)
	относительная адресация через локальную переменную и индекс rDp = [sp[imm-loc]] + rSi; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
ari (address from register and immediate)	r0/r1, imm, r0/r1	pointer/все регистры	2	2c8 (712)
	относительная адресация через регистр и индекс – константу rDp = [rDp]+ imm; rDp = {r0p, r1p}			
arr (address from register and register)	r0/r1, r0/r1i, r0/r1	pointer/все регистры	2	2ca (714)
	относительная адресация через регистр и индекс rDp = [rDp]+ rSi; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
vgi (value addressed by global and immediate)	imm-glob, imm, r0/r1	все типы/все регистры	12	258 (600)
	взятие значения адресованного глобальной переменной и индексом – константой rDp = [[[imm-glob]]+ imm]; rDp = {r0p, r1p}			

vgr (value addressed by global and register)	imm-glob, r0/r1, r0/r1	все типы/все регистры	12	264 (612)
	взятие значения адресованного глобальной переменной и индексом rDp = [[[imm-glob]]+ rSi]; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
agi (address from global and immediate)	imm-glob, imm, r0/r1	pointer/все регистры	2	2c0 (704)
	относительная адресация через глобальную переменную и индекс – константу rDp = [[imm-glob]] + imm; rDp = {r0p, r1p}			
agr (address from global and register)	imm-glob, r0/r1i, r0/r1	pointer/все регистры	2	2c2 (706)
	относительная адресация через глобальную переменную и индекс rDp = [[imm-glob]] + rSi; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
ptl* (pointer to local)	imm-loc r0/r1	pointer/все регистры	2	2f8 (760)
	взятие адреса локальной переменной rDp = sp + imm-loc; rDp = {r0p, r1p}			
ptg* (pointer to global)	imm-glob r0/r1	pointer/все регистры	2	2fa (762)
	взятие адреса глобальной переменной rDp = imm-glob; rDp = {r0p, r1p}			
vll* (value addressed by local)	imm-loc r0/r1	все типы/все регистры	12	2fc (764)
	взятие значения адресованного локальной переменной rDp = [sp[imm-loc]] ; rDp = {r0p, r1p}			
vlg* (value addressed by global)	imm-glob r0/r1	все типы/все регистры	12	308 (776)
	взятие значения адресованного глобальной переменной rDp = [[imm-glob]] ; rDp = {r0p, r1p}			
esi*	r0/r1, imm, r0/r1	pointer/все регистры	2	2d2 (722)
	взятие адреса через базовый регистр и индекс - константу rDp = [rDp] + imm; rDp = {r0p, r1p}			
esr*	r0/r1, r0/r1, r0/r1	pointer/все регистры	2	2d0 (720)
	взятие адреса через базовый регистр и индекс rDp = [rDp] + rSi; rDp = {r0p, r1p}; rSi = {r0i, r1i}			
ehtable (exception handlers table)	imm		1	339 (825)
	устанавливает таблицу контекстных обработчиков прерываний handlers_table = [imm]			
epop (exception object pop)			1	33a (826)
	загружает из стека значение активного исключения. Извлекается объект типа pointer который должен являться указателем на объект, описывающий исключений pop raised_exception			
raise			1	33b (827)
	проверка наличия активного исключения (не ноль в регистре raised_exception) и поиск подходящего обработчика исключений			
signal			1	33c (828)
	проверка наличия активного исключения (не ноль в регистре raised_exception) и выбрасывание исключения в контекст вызывающей подпрограммы			
is	imm	integer/все регистры	2	33d (829)
	извлекает указатель на объект из стека (тип pointer) и проверяет, реализует ли данный объект тип с именем [imm]. Значение [imm] должно указывать на С-строку.			
cfcall (concurrent	imm		1	340 (832)

	вызов сопрограммы по непосредственному адресу			
mli* (move into local from immediate)	imm, imm-loc	все типы/без рег.	2	341 (833)
	помещение константы в локальную переменную			
cast	imm		1	33f (831)
	извлекает указатель на объект из стека (тип pointer) и преобразует его к типу с именем [imm]. Если данный объект не реализует тип, к которому происходит преобразование, то выбрасывается исключение. Значение [imm] должно указывать на C-строку.			

* эти семейства инструкции не используются текущей версией компилятора

Таблица 6. Вспомогательные инструкции (инструкции настройки адресов)

label	id		1	5 (5)
	метка в коде. При предкомпиляции номера меток заменяются их абсолютными значениями.			
const	type, size, data		1	4 (4)
	Позволяет хранить константы в теле программы, а так же задает ряд операций по предварительной настройке программы, таких как позднее связывание и настройка адресов.			

Таблица 7. Операции предкомпиляции инструкции `const`

название типа константы	назначение
RAWDATA	Хранение однотипных массивов данных, например строк. Данные преобразуются в формат, требуемый исполняющей платформой, адрес метки становится равной области хранящей подготовленные данные.
VMLT	Подготовка структуры описывающей объекты определенного класса.
BINDING	Привязка к функции из динамической библиотеки в платформенно-зависимом формате. Адрес метки становится равным адресу соответствующей функции в динамической библиотеке.
DYNLIB	Добавление библиотеки в платформенно-зависимом формате к списку библиотек используемых директивой BINDING.
PACKAGE	Открытие, компиляция и инициализация Лемик-пакета.
IMPORT	Импорт функции из указанного Лемик-пакета.
IMPORT_GLOBAL	Импорт глобальной переменной из указанного Лемик-пакета.
EHTABLE	Подготовка структуры описывающей контекстные обработчики прерываний используемые в данной функции.

Некоторые инструкции (а именно `vcall`, `icall`, `is`, `cast`, `raise`, `signal` и `objvi`) ожидают особую структуру объекта в памяти, с которым они работают. Эта структура (см. рис. 1) несет описание объекта – перечисление классов и интерфейсов, реализуемых данным объектом, и список адресов виртуальных методов.

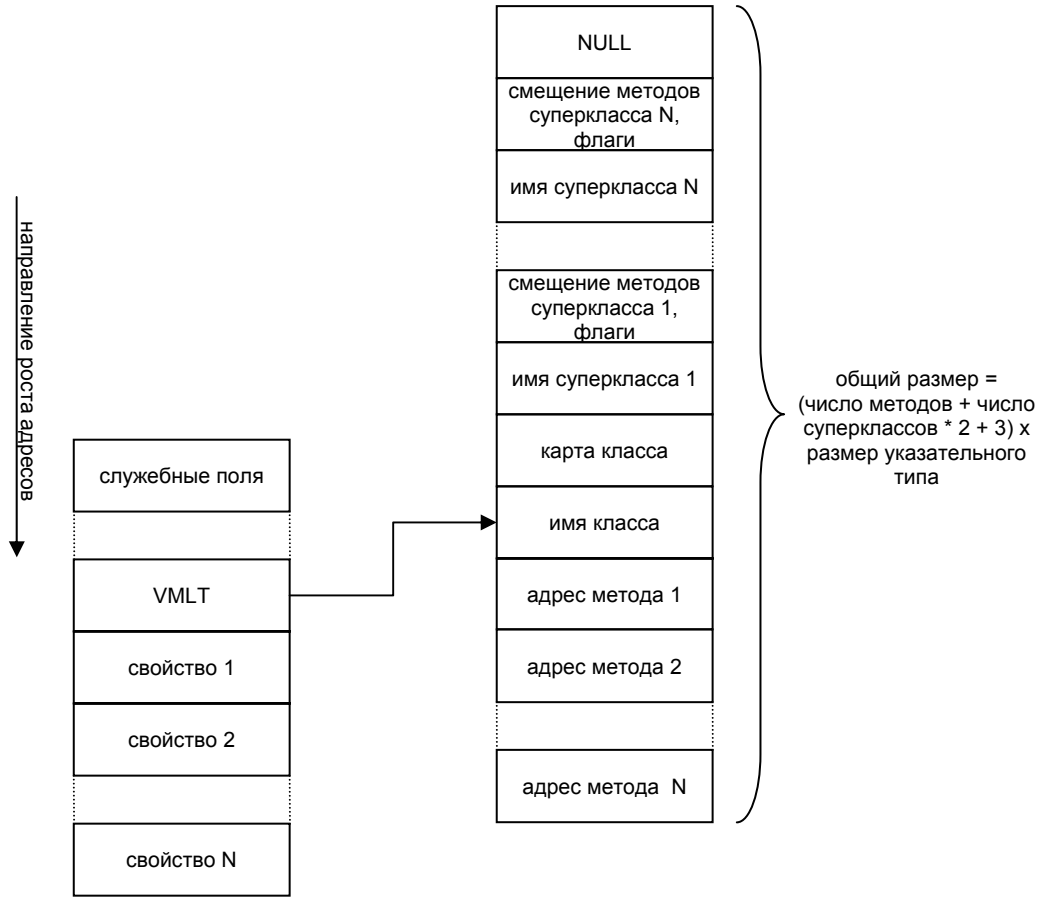


Рис. 1. Формат объекта с описывающей структурой

Рассмотрим, например, как происходит вызов виртуального метода класса.

Таблица 8. Вызов виртуального метода

исходный текст программы	ассемблер виртуальной машины для главной функции

`class Foo`
`  private a as Integer`
`  public sub meth: a = 1 : end sub`
`end class`
`class Foo2 extends Foo`
`  private b as Integer`
`  public sub meth: a = 2 : end sub`
`end class`
`dim e as Foo = new Foo2 : e.meth`
	`.start-segment main`
	`  enter  #96, 0`
	`  objvi  L#102, 12, r0p`
	`  push  r0p`
	`  vcall  L0`
	`l96:`
	`  leave`
	`.end-segment`

Операция `objvi L#102, 12, r0p` создает новый объект размером 12 байт с таблицей описания объекта адресованной меткой L#102. Адрес объекта помещается в регистр r0p. Инструкция `push r0p` помещает созданный объект на стек, так как это первый (хотя неявный) и единственный параметр вызываемого метода. Метод вызывается косвенно, через таблицу описания объекта и его номер в таблице - `vcall L0`. Реальный адрес метода находится в таблице со смещением 1 (номер метода + 1).

add, 4